Proyecto de laboratorio MIPS

Lucas Mesias (3 horas)
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, Santiago, Chile
lucas.mesias@usach.cl

***Resumen*—Se asignó la tarea de crear una libreria de operaciones matemáticas vectoriales programada en MIPS para un sistema de navegación de una nave subacuatica no tripulada, el cual trabajará con las coordenadas [X, Y, Z] de la nave. Para lograr este objetivo, se aprenderán conceptos básicos de la programación en MIPS para lograr implementar las operaciones requeridas.**

***Palabras claves*—Programación, aprender e implementar.**

I. Introducción

La tarea encomendada es crear un módulo lógico matemático para ser implementado en el sistema de navegación de una nave subacuática no tripulada, para el cual se debe implementar una librería en MIPS para la manipulación de vectores, ya que la navegación de esta nave está basada en las coordenadas tridimensionales (x, y, z), para esto se programará la solución en el simulador MARS para MIPS. El objetivo es crear una librería usable, que sea compatible con el formato requerido y que ejecute correctamente.

II. Antecedentes

Aunque las operaciones requeridas son simples de realizar con lápiz y papel, implementarlas en MIPS no será tan sencillo.

II-A. Sintaxis

En primer lugar se debe comprender la sintaxis de MIPS, donde cada línea de código contiene una instrucción, seguida de los parámetros que esta requiere, donde la mayoría del tiempo, estos serán registros y direcciones de memoria.

$$ADD\ \$a0,\ \$t0,\ \$t1 \tag{1}$$

Lo que significa sumar el contenido del registro t0 y t1, y guardar el resultado en el registro a0.

II-B. Separación de codigo

Luego, se deben entender las secciones del código, ya que está separado por 2 partes, la primera es la sección .data, en la cual se encuentran los datos de memoria que se usarán en el programa, en este caso se reserva la memoria donde se encontrarán los números con los que se va a trabajar con .space. También se almacenan los mensajes de salida para organizar el código.

```
.data
        m: .space 108
        texto1: .asciiz "X: "
        texto2: .asciiz "Y: "
        texto3: .asciiz "Z: "
        espacio: .asciiz " "
```

En segundo lugar, la sección .text contiene el código/programa.

```
.text
        # Cargar vector M en registro
        la $s0, m($zero)

        # X1 + X2
        li $v0, 4
        la $a0, texto1
        syscall
```

II-C. Memoria vs Registro

Finalmente, se debe entender la diferencia entre memoria y registro, ya que la memoria se usa para almacenar información, pero para trabajar con esta, se debe llevar la información a los registros, los cuales son otro tipo de memoria de mayor velocidad, con la cual el procesador puede trabajar directamente, una vez que los resultados son calculados, la información se vuelve a guardar en la memoria.

III. Materiales y métodos

III-A. Materiales

Se hace uso de MARS 4.5 (MIPS Assembler and Runtime Simulator) para programar.

III-B. Métodos

Las operaciones requeridas fueron desarrolladas en papel, para ordenar los pasos requeridos para completar las mismas. En cada implementación se repiten varias veces los mismos conjuntos de instrucciones:

- Imprimir texto

```
li $v0, 4
la $a0, texto1
syscall
```

- Cargar valores desde memoria a registros, realizar una operación e imprimir el resultado

```
li $v0, 1
lw $t0, ($s0)
lw $t1, 20($s0)
add $a0, $t0, $t1
syscall
```

- Pedir un valor desde consola

```
# Obtener S
li $v0, 4
la $a0, multPrompt
syscall

li $v0, 5
syscall
move $s1, $v0
```

Con diferentes combinaciones de estos bloques de códigos, entre otras instrucciones, se pueden realizar todas las operaciones requeridas.

IV. Resultados

Operación 1:

x=1+4, y=2+5, z=3+6
x: 5, y: 7, z: 9

Operación 2:

x = 1, y = 2, z = 3, s = 2
x: 2, y: 4, z: 6

Operación 3:

x1 = 1, y1 = 2, z1 = 3, x2 = 4, y2 = 5, z2 = 6

```
Ingrese X: 1

Ingrese Y: 2

Ingrese Z: 3
X: 4 Y: 10 Z: 18
```

V. Conclusiones

Se implementaron las operaciones no opcionales solicitadas con éxito, resueltas con algoritmos simples, lineales y usando solamente entre 1 y 3 registros para resolver las operaciones, aunque específicamente el programa 3 podría usar aún menos, no es un requerimiento optimizar el programa para usar la menor cantidad de registros.